Techniques of Assessment Pertinent to Computer Programming Courses

Peter P. Smith
College of Science
Buena Vista University
610 West Fourth Street
Storm Lake, IA 50588
smithp@bvu.edu

Abstract

Computer programming courses present unique challenges with regard to assessment. This is primarily due to the fact that students in such courses must be able to demonstrate competence in two distinct areas, two separate modes of functioning: they must be able to understand and verbalize the *concepts* involved in computers and programming, and they must also be able to produce well written, well structured and understandable *applications* in the language involved. Although the two are complementary, it is nonetheless required that two separate skill sets be evaluated.

We have developed a schedule of evaluations for programming courses that allows instructors to gauge the progress of students in both of these areas as the course proceeds. By the end of the course both instructor and students have a very good idea of how well students have developed a mental construct of computers and programming, and how well students have developed programming skills and techniques that allow them to write functioning applications.

Introduction

Courses in computer programming present unique challenges with regard to teaching, learning, and assessing student progress. This is primarily due to the fact that students in such courses must be able to demonstrate competence in two distinct areas, two separate modes of functioning: they must be able to understand and verbalize the *concepts* involved in computers and programming, and they must also be able to produce well written, well structured and understandable *applications* in the language involved. Although the two are complementary, they nonetheless comprise two separate skill sets.

The traditional approach to assessing student understanding and abilities involves a combination of programming assignments and written exams. Typically instructors give out-of-classroom assignments intended to test student program writing abilities, and they give in-class exams to test how much students understand of the concepts. These latter exams are also frequently used as an additional check on language syntax. The problem with this approach is that it does not delineate the two skills very well. This is because, on the one hand, the instructor can never be sure exactly who wrote the programming assignments, and on the other hand because programming aspects are mixed in with the written exams. For example it's not uncommon to ask questions about the nature of control structures right along with questions about control structures in the language of interest. This makes it difficult to tell whether the students know either.

This inability to differentiate skill levels is especially important in introductory courses of computer science. Quite often students get through a course with a decent grade but still have problems with actual programming.

Another major problem concerns the setting and environment in which out-of-class programming assignments are done. It is tempting, and easy, for students to get assistance in writing these programs. In fact plagiarism and cheating are fairly common in such situations. Consequently, since the actual author of a given program is in doubt, the final application may not be at all indicative of the skill level of the student who hands it in.

We have examined these difficulties via the general education offering in computer science at Buena Vista University. Although the course under discussion is intended for non-majors, it does require students to study both the underlying principles of computer science *and* methods of writing applications. The language used is Visual Basic. We describe here methods designed to address these concerns. Additional preliminary results also suggest that these methods are applicable to the CS1 course for prospective majors.

Motivation

The motivation for this study stemmed from problems that arose with regard to out-of-class programming assignments. There was a great deal of plagiarism, copying, and academic dishonesty in general occurring, and some method of mitigating this was needed. In addition, at about the same time, Buena Vista University embarked on a concerted effort to build assessment into the curriculum. The goal was to make assessment an integral part of the course rather than having it a separate add-on responsibility. This involved, among other things, finding ways to implement performance based outcomes such that student competence could be established at the conclusion of a course.

Methodology

In order to address the above stated concerns, we began a schedule of activities intended to assess the two separate skill sets and at the same time reduce the motivation for, and incidence of, academic dishonesty. Three kinds of activities are used: Assignments, Practical Exams, and Concepts Exams.

Assignments

These are typical outside-of-class programming assignments. They are progressive and increase in difficulty as the semester proceeds. The purpose is, of course, to give students the opportunity to implement techniques of programming explained in class, and to make it possible for them to practice these skills in a variety of contexts. In addition they reinforce many of the computer concepts taught during lecture. As we implement the strategy here, assignments may be done in groups, and students are permitted to utilize any resources they wish provided only that they acknowledge such outside assistance. Since these assignments contribute minimally to the final course grade for a student, there is less incentive to cheat in order to get them done correctly. Table 1 shows typical programming assignments given during the semester.

Personalized *Hello World* Application
Trip Computer
On-Line Shopping
Mortgage Loans
Complete Payroll Application
Cash Register Simulation
Traffic Light Controller
Acey Deucy
File Based Payroll
Contact List

Table 1: Typical Programming Assignments

Practical Exams

Practical Exams, on the other hand, require students to write applications in a controlled environment. Students must work individually; they are permitted no consultation with other people at all. However, they may use any other kinds of resources they want, including notes, web pages, and previous assignments. The Practical Exams require techniques and methods similar to those needed to complete the antecedent assignments. Students are thus encouraged to learn these techniques via the assignments since they will have to use them when they take the Practical Exams. Because the Practical Exams *do* contribute substantially to final grades, there is strong motivation to do well on them. Table 2 shows some typical applications students must design for Practical Exams.

Railway Ticket Prices
Simplified Atomic Spectra
Gas Pressures
Fibonacci Numbers
Clock Timer
Quadratic Equations
Simplified Database
Course Letter Grades

Table 2: Typical Applications Required for Practical Exams

Concepts Exams

Finally, students must take traditional written exams covering the concepts and models of computer science. In the Visual Basic course this is done via on-line objective tests. The exams include True/False, multiple choice, and completion questions. Although the Concepts Exams are not intended to assess programming skills or language syntax, they sometimes may include various elements of Visual Basic in order to reinforce some concepts, for example, data types. Table 3 shows the kinds of questions that are asked of students on the Concepts Exams.

- *A description of a process* is the definition of -----.
- An asynchronous occurrence that initiates the execution of a procedure is called a(n) -----.
- Taking *data* and converting it to *information* is termed the ----- model.
- What is *operator precedence?*
- The Fundamental Control Structures are said to be *fundamental* because
- The process of finding and eliminating errors in a program is called -----.
- One of the motivations for using subroutines is that they allow you to break up one large task into several smaller tasks. The term for this is -----.

Table 3: Typical Questions Asked on the Concepts Exams

Timing of Instruments

Assignments are given approximately once a week. Practical and Concepts Exams are given four times during the semester in the form of three semester exams and one comprehensive final exam. This spreads out the load and doesn't concentrate too much material in one place and at one time.

Results

Table 4 shows the results of data obtained over several years of teaching the Visual Basic Programming course. These results tend to support our original assertions concerning the difficulties inherent in assessing outcomes in programming courses. First, there do indeed seem to be two separate skill sets involved. Second, there is not a very high correlation between student results on lab assignments and student results on the Practical Exams. This in turn implies that lab work is not a very good indication of student programming competence.

Assignment Average	80.5
Practical Exams Average	78.3
Concepts Exams Average	79.5
Correlation between Assignments and Practical Exams	0.44
Correlation between Practical Exams and Concepts Exams	0.58

Table 4: Summary of Data Collected for Visual Basic General Education Class

Skill Sets

The results indicate that two separate skills are extant. The correlation between scores on the Practical Exams and Concepts Exams, 0.58, is not very high, and is mild at best. This appears to be reasonable: although knowing one can help with the other, and knowing both provides a synergistic effect, they are still separate activities. Being good at one does not necessarily transfer to being good at the other.

Lab Work as an Indicator of Programming Competence

Of greater interest is the fact that student performance on the Practical Exams, which is carried out under controlled conditions, is very poorly correlated with student performance on the programming assignments: the correlation is only 0.44. The coefficient of determination is $0.44^2 = 0.19$, which means that only 19% of student performance on the practical exams can be explained by what they seem able to do on lab assignments.

These two results taken together tend to suggest that the traditional way of assessing overall computer competence is lacking. Trying to measure the two skill sets via lab assignment and written exams which also cover programming will fall short. Lab work and written exams seem not to be a very good way of determining whether or not students understand the concepts *and* can actually write programs.

Additional Observations

The incidence of cheating appears to have decreased markedly after implementation of this assessment regime. Students now seem to be using the lab assignments for their intended purpose, namely, as a way of learning the skills that will be necessary for the Practical Exams. Cheating on the homework may cost them seriously in their final grade, not because it will be noted and sanctioned, but rather because it will not enable them to do well on exercises that strongly affect their final grades.

Informally we have also found that the final grade awarded to a student is usually pretty close to what an instructor intuitively feels a student will get based on in-class performance and participation.

Discussion

This paradigm and exercise schedule seems to work very well for the Visual Basic programming course. It allows differentiation of students by both skill level and skill type, and allows counseling to be better focused on the student's area of difficulty. One can tell pretty easily whether or not a student is doing the assignments on his or her own simply by examining the difference in performance on the Practical Exams. A student cannot easily get a free ride through the assignments. An added benefit is that it allows early identification of outstanding students in the field.

On a practical level these assessment techniques reduce the amount of effort required on the part of an instructor with regard to grading. Since there is reduced motivation for cheating on lab assignments, much less effort is required to look for signs of plagiarism and copying. It also encourages students to work together cooperatively.

This situation is quite similar to that found in real life. For example, although it is not necessary to understand the functioning of a bicycle in order to ride one, nor is it necessary to be able to ride one in order to understand how it works, knowing both gives one an advantage in either aspect.

Finally, preliminary results suggest that the techniques described here can be usefully applied to the major's sequence CS1/CS2. It would also be interesting to see if the same holds true for upper division courses.

Conclusions

These techniques appear to be an efficacious way of assessing student performance in the two areas important to computer science, namely, understanding the concepts of computer science and program writing ability. It provides a mechanism for assessment, and can be used to demonstrate competence on the part of individual students. We will continue to use this methodology and extend it to upper division courses to see if it holds there as well.

References

ACM Computing Curricula 2001. (2001). *Chapter 7 Introductory Courses*. Retrieved March 5, 2005, from http://www.computer.org/education/cc2001/final/chapter07.htm.

Chamillard, A. T., Braun, K. A. (2000). *Evaluating Programming Ability in an Introductory Computer Science Course*. Retrieved March 4, 2005, from http://delivery.acm.org/10.1145/340000/331857/p212-chamillard.pdf?key1=331857&key2=6957750111&coll=GUIDE&dl=GUIDE&CFID=39866760&CFTOKEN=57637402.

Day, C., Waldron, J. (2004). *Assessing the Assessment of Programming Ability*. Retrieved March 8, 2005, from http://db.grinnell.edu/sigcse/sigcse2004/viewAcceptedSession.asp?sessionType=Paper&sessionNumber=166.

McCracken, M. (2001). *Assessment of Programming Skills of First Year CS Students: Do they know how to program?* Retrieved March 9, 2005, from http://www.cc.gatech.edu/projects/iticsewg/csas_proposal.pdf.